RDFa2: Lightweight semantic enrichment for hypertext content

Bai, X., Klein, E., & Robertson, D. (2011). RDFA2: Lightweight Semantic Enrichment for Hypertext Content. In The Semantic Web. (pp. 318-333). (Lecture Notes in Computer Science). Springer Berlin / Heidelberg. 10.1007/978-3-642-29923-0_21

RDFa^2: Lightweight Semantic Enrichment for Hypertext Content

Xi Bai, Ewan Klein, and Dave Robertson

School of Informatics, University of Edinburgh, UK
xi.bai@ed.ac.uk, ewan@inf.ed.ac.uk, dr@inf.ed.ac.uk

Abstract. RDFa is a syntactic format that allows RDF triples to be integrated into hypertext content of HTML/XHTML documents. Although a growing number of methods or tools have been designed attempting at generating or digesting RDFa, comparatively little work has been carried out on finding a generic solution for publishing existing RDF data sets with the RDFa serialisation format. This paper proposes a generic and lightweight approach to generating semantically-enriched hypertext content by embedding RDF triples derived from diverse provenances in terms of a concept of topic nodes which will be automatically recommended by our discovery algorithm. RDFa^2 is a proof-of-concept implementation for our approach and works as an online platform assisting Web content publishers in semi-automatically generating, personalising and curating pages with RDFa. RDFa^2 has been introduced and employed by students in a master level course and the experimental results as well as additional case studies indicate the validity of this approach to generating triple-embedded Web documents such as online profiles and vocabularies with little user intervention.

1 Introduction

The Semantic Web have been proposed as an extension of the Document Web where human-readable hypertext documents currently dominate. As part of the Semantic Web initiative to promote machine-readability of Web documents, RDFa (a W3C recommendation for more than two years) has been designed so that "authors can markup human-readable data with machine-readable indicators for browsers and other programs to interpret" [1]. An increasing number of tools for processing RDFa have been developed which leverage the existing range of techniques for processing standard RDF. Hundreds of thousands of FOAF [1] documents have been created (semi-)automatically or manually but due to the lack of human readability of RDF triples, many of RDF documents are hidden in repositories or behind SPARQL endpoints which are not accessible to users without expertise. On the other hand, a growing number of ready-to-reuse Linked Data sets have been published nowadays and a fully-fledged Linked Data application is likely to make use of data from more than one source. Automatic information processing and integration hence require Web content to be

[1] http://xmlns.com/foaf/spec

not only human-readable but also machine-readable. Although content publishers can publish a plain HTML page and point it via `meta` to an RDF document, since they are separate documents the availability of both documents may not be achieved at the same time and it is also difficult to avoid data duplication. While RDFa makes it easy for Web authors to manually add small amounts of semantic markups to XHTML documents, RDFa also offers the potential to transform pre-existing machine-readable data into human-readable format. So far, this has received relatively little attention. We propose a generic and lightweight approach to semi-automatically generating semantically-enriched hypertext content from existing RDF documents [2]. A key ingredient, which we will describe in more detail below, involves the identification of one or more "topic nodes" in the RDF context(s) to guide the injection of RDFa into an XHTML template. A proof-of-concept of this approach has been implemented under the name RDFa^2 (*RDFa a*nnotator), and has been available as an online service [3]. RDFa^2 runs within standard Web browsers, and allows users to customise its output in two ways: either by modifying the generated data in an edit window (with on-the-fly preview) or by revising the generated XHTML template, which can be saved to local storage for future use.

The remainder of this paper is organised as follows. Section 2 reviews related work on processing RDFa. Section 3 describes the preprocessing of "RDF contexts" required by our approach. Section 4 proposes a hybrid topic-node discovery method based on weighted occurrences of nodes as well as heuristic properties and details how our approach can assist users in creating, customising and reusing Web content with RDFa. How RDFa^2-assisted data integration is compliant with the standard RDF data model as well as the Linked Data [8] principles is also discussed in this section. Section 5 evaluates this approach by introducing our prototype to students in a master level course as well as case studies on republishing online vocabularies. Section 6 draws conclusions and indicates future work.

2 Related Work

SPARQLScript [4] supports output templating which allows users to embed SPARQL query results into the dynamic generated Web pages via place holders. It could be used for dynamically embedding triples and users however need to learn this PHP-like script language and SPARQL. Fresnel [16] is a declarative language for rendering RDF content in specific browsers (as of writing this paper, it supports five browsers) but requires users to learn how to write lenses and formats which are two foundational concepts employed in this language. Tal4Rdf (T4R) [5] is a template language for presenting RDF data into other formats such

[2] Here and in the rest of the paper, we take "RDF document" to subsume any documents containing (or embedding) RDF triples.

[3] `http://demos.inf.ed.ac.uk:8836/rdfasquare`

[4] `https://github.com/semsol/arc2/wiki/SPARQLScript`

[5] `http://liris.cnrs.fr/~pchampin/t4r/`

as HTML, SVG, JSON, Atom and so on. However, it currently does not support the XHTML+RDFa representation of existing RDF data so it is difficult if not impossible for other users or developers to repurpose (e.g. mashup) the reformatted data on the generated Web pages. Therefore, although the above designed languages dedicated to RDF-embedded page generation are self-adaptive to updating of triples, the cost of creating an appropriate intermediate format (e.g., templates or lenses) is not much less than the cost of manually creating an XHTML+RDFa page.

FOAFr [6] allows users to convert their FOAF documents into XHTML pages with RDFa automatically and it is however focused on the FOAF vocabulary [7] only. Likewise, *FOAF.Vix* [8] is a visualiser and relation explorer for FOAF documents. It provides RDF documents serialised in RDF/XML and Web pages containing RDFa with visualisations in which there is no embedded meta information. *GoodRelations* [13] provides a *GoodRelations Annotator* [9] as well as a *Rich Snippet Generator* [10], both of which assist users in creating RDFa snippets for their businesses or products using the particular *GoodRelations* vocabulary. By filling slots in a provided template, a user will get an RDFa snippet generated using XSLT. Our approach is not domain specific and allows users to generate RDFa snippets using any vocabularies in the RDF data model.

RDF2RDFa [14] also allows users to copy and paste RDFa snippets generated from input RDF documents. This copy-and-paste method makes the original RDF content transparent to users so it is difficult if not impossible that users can reuse human-readable content from the original RDF documents. *Drupal 7* allows developers to generate templates for associating RDFa with *Drupal* elements such as content types and fields [10]. It has, however, not offered a fine-grained solution for content publishers to easily associate RDFa with more open content lacking a generalised template. Our RDFa^2 provides users with the free-editing functionality and on-the-fly previews after content change so they can get the real WYSIWYG experience when starting the transformation.

3 Topic Nodes and Topic Trees

Our algorithm for transforming RDF documents to XHTML+RDFa pages is based on automatically generated templates. These templates are schematic XHTML documents, and have a tree structure. By contrast, the RDF data model is a graph, and cannot be converted to a single tree without duplicating re-entrant nodes. In order to overcome this problem, the conversion from RDF requires users to select a specific node in the RDF graph which then forms the root of a tree of RDF statements. Which node should the user choose? In practice, this seems to follow straightforwardly from the user's goals, namely to

[6] `http://sw.joanneum.at:8080/foafr`
[7] `http://xmlns.com/foaf/spec`
[8] `http://foaf-visualizer.org`
[9] `http://www.ebusiness-unibw.org/tools/goodrelations-annotator/en`
[10] `http://www.stalsoft.com/grsnippetgen`

focus on the resource which is his or her main topic of interest in the resulting XHTML page. For example, in the case of a FOAF file, the obvious resource to choose is the value of the `maker` or `primaryTopic` property.

The node that is targeted in this way is called the *topic node*. The RDF document from which the topic node is derived is called the *RDF context*, and relative to a context $\mathcal{C}$, a set of RDF statements rooted in a topic node is called a *topic $\mathcal{C}$-tree*. We distinguish between two kinds of topic trees, depending on the position (the subject or the object) of the topic node inside a specific triple. Given a resource r, context $\mathcal{C}$, and RDF statement (s, p, o), the *subject (topic) $\mathcal{C}$-tree based on r* is defined as $\{(s, p, o) \in \mathcal{C} \mid s = r\}$, and similarly for the *object (topic) $\mathcal{C}$-tree based on r*.

The notion of a topic tree for a topic node is essentially the same as a *bounded description* of a resource; that is, where "a sub-graph can be extracted from a data set which contains all of relevant properties and relationships associated with a resource" [11]. For the sake of clarity, a topic node is not necessarily the global topic of an RDF document; rather, it corresponds to a resource in the document which the user regards as interesting enough to represent in XHTML. Figure 1 illustrates the selection of a subject topic tree from an RDF context.

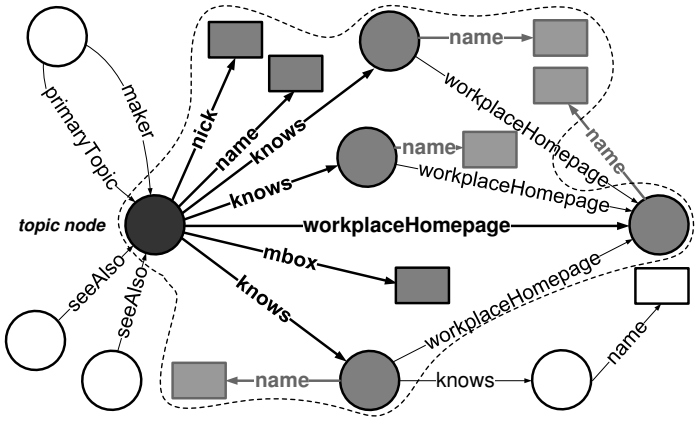


Fig. 1. *Subject (topic) $\mathcal{C}$-tree* of a FOAF document

In this figure, for the sake of brevity, we have omitted the name spaces (henceforth abbreviated as NS) of all properties here. In this figure, circles denote resources and squares denote literals. The node coloured in dark grey is the current topic node while the sub-graph surrounded by the dashed line is the subject topic tree for this node. The labelling information about the resources in the subject position are also included in the topic tree in order to make the resources themselves human-readable on the RDF-embedded page.

Although the most straightforward use case for our approach creates a standalone XHTML page from an RDF document, we also want to accommodate

cases where the output of this approach is inserted as a snippet into a larger (X)HTML document. Taking Sir Tim Berners-Lee's Twitter profile as an example, Figure 2 illustrates RDFa^2 generated an RDFa snippet from the triples obtained via SemanticTweet APIs [11] (it is needless to mention FOAF documents can be fed into this tool directly as snippet-generation seeds [3]). In the copy&paste way, this snippet is ready to be inserted into the `<body>` section of the homepage or exported as a separate Web page and notably, it can be further customised by publishers through adding more human-readable content.

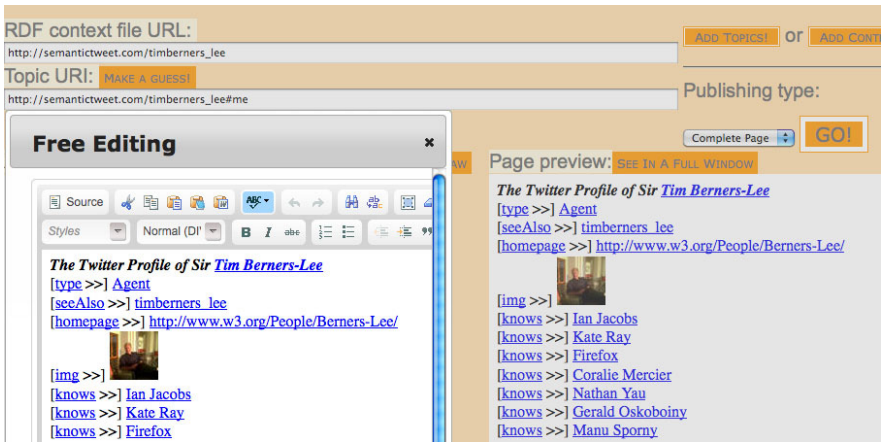


Fig. 2. Personalise the raw page generated by RDFa^2

4 Embedded-Annotation Generation

Our approach to assisting users (e.g., Web content publishers) in generating annotations embedded in their hypertext content is detailed in this section. This approach has the ability to automatically discover a candidate set of topic nodes (from existing RDF contexts) which can be offered to the user thereafter and also supports federated integration in the sense that users can embed multiple topic nodes from multiple RDF contexts into a single Web page. Within the publishing process, publishers can revise suggested annotated blocks or raw pages in terms of their individual requirements. Moreover, templates are also provided via our approach and customised by publishers (and also stored, loaded and reused) if needed. Algorithm 1 describes the annotation generation process (generating the partial snippet for the subject topic tree) and will be further discussed in the following subsections. Likewise, the snippet generation corresponding to the object topic tree is not described here due to the space limitation but can be

[11] `http://semantictweet.com/`

achieved by revising this algorithm and moving the topic node from the subject position to the object position. For the mashup purpose, the embedded RDF triples can be harvested and serialised in several formats such as Notation3 (N3) [7], RDF/XML [6], N-Triples [12] and Turtle [5].

Algorithm 1: RDFa Snippet Generation Algorithm (subject (topic) $\mathcal{C}$-tree)

Input: *topic_uri*, the URI of the topic node and *model*, the model containing triples in the current context.
Output: *rdfa_snippet*, the RDFa snippet representing the information about the inputted topic node.

```
begin
    def rdfa_snippet = getDIVHead(topic_uri);
    def sub_topic_tree = model.getStantementsBySubject(topic_uri);
    def properties = model.getUniquePropertiesBySubject(topic_uri);
    for each property in properties do
        def objects = sub_topic_tree.getObjectsByProperty(property);
        def prop_local_name = property.getLocalName();
        def prop_node_name = (property.getNameSpace() + "_" +
        prop_local_name + "rel").replace("_", "dash");
        def prop_curie_name = model.getPrefix(property.getNameSpace()) +
        ":" + prop_local_name;
        for each object in objects do
            if object.isListeral() then
                rdfa_snippet += "<#if topic." + prop_node_name + "??>" +
                "<#list topic." + prop_node_name + "?keys as key>" +
                getLiteralStyle(prop_local_name, property.getURI()) + ... ;
            else
                def snippet = "";
                if object.isURIResource() && object.getURI().indexOf(".") !=
                -1 then
                    def obj_uri = object.getURI();
                    def expansion =
                    obj_uri.subString(obj_uri.lastIndexOf("."));
                    snippet += getSnnipetByExpansion(prop_curie_name,
                    prop_node_name);
                else
                    snippet += "<a rel='" + prop_curie_name + "'
                    href='${topic." + prop_node_name + "[key].uri}'
                    onclick='return false;'>${topic" + prop_node_name +
                    "[key].uri}</a></span><br/>";
                rdfa_snippet += "<#if topic." + prop_node_name + "??>" +
                "<#list topic." + prop_node_name + "?keys as key>" + "<#
                if topic." + prop_node_name + "[key].uri??>" +
                getResourceStyle(prop_local_name, property.getURI(), true) +
                snippet + "<#if><#list><#if>";
    return rdfa_snippet;
```

4.1 Topic-Node Discovery

In the preceding section, we assumed that topic nodes will be selected by the user. However, this requires the user to understand the basic syntax of the RDF context inside which these node are represented. One way of automatically identifying topic nodes in a given RDF context is to query the document for URIs with properties that are diagnostic of topic-hood, such as `foaf:primaryTopic` or `foaf:maker` in FOAF files. However, not all RDF documents contain such properties, and even in FOAF files which do employ them, they do not always take semantically appropriate values. Consequently, topic nodes cannot reliably be detected just in terms of the semantics of statements in the RDF context itself. Xiang et al. compared five measurements from three categories (degree centrality, shortest-path-based centrality and eigenvector centrality) for automatically summarising ontologies in a topic-independent manner and their interesting evaluation showed that weighted in-degree centrality measures and several eigenvector centralities all have good performance on ontology summarisation [17]. As analysed in [4], for the case that the target RDF documents mix up ontology-related triples and individual-related triples, the above topic-free measurements may be affected by unforeseen noise nodes. Moreover, each property could have a corresponding inverse property so it is difficult if not impossible to draw a conclusion that an RDF node's in-degree (or out-degree) prioritises its out-degree (or in-degree). In this paper, we propose an improved algorithm for semi-automatically discovering and recommending topic nodes. Since the RDF data model is a directed graph and nodes are connected to one another through directed edges, one solution for discovering the topic node is based on node connectivity. In other words, the more edges (outgoing or incoming) a node has, the more important it is likely to be. In order to maximise the accuracy of this heuristic, our algorithm selects the top n most highly connected URIs and offers them to users for subsequent confirmation [12]. Perhaps not surprisingly, this algorithm works especially well for RDF documents such as FOAF files that usually do have a central topic.

When a user inputs the URI of a resource that she wants to integrate into her Web page, together with an RDF context, RDFa^2 will query this context with the selected URI for all statements in which the URI is either subject or object. From this set, a subject (respectively, object) topic tree will automatically be selected if it exists. Its root will be the topic node and its corresponding properties and values will be stored in other nodes or leaves. Then the user can refer to any information about this topic node using the path structure `root.predicate.values[key].[resource]` or `root.predicate.values[key].[literal]` in the template which will be discussed in Subsection 4.3. Here, `root` denotes the resource currently being integrated; `predicate` denotes a specific property with which this resource is associated; and `values` is a list that stores the values of a property (since some properties may have multiple values). The

[12] The value of n can be any reasonable integer. Although RDFa^2 takes n to 10, by default it only just shows the top three URIs to users. It is also worth noting that blank nodes are filtered out from the set of candidates.

screenshot in Figure 3 illustrates how topic nodes derived from an RDF context (Sir Tim Berners-Lee's twitter profile in RDF) were discovered and the most important URI in this context was shown at the top of the recommendation list.

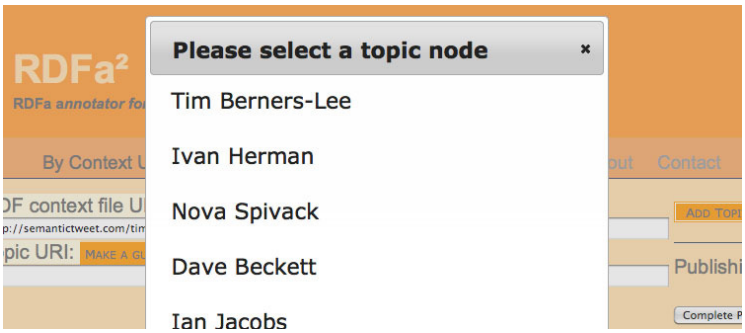


Fig. 3. Screenshot for discovering the topic node from an RDF context

4.2 Federated-Annotation Generation

We do not want to exclude the possibility of the user selecting more than one topic node from a given RDF context. For example, a user may wish to render the FOAF document vocabulary (i.e., encoded as a set of RDF statements) as XHTML, and in this use case, all of the nodes `foaf:Person`, `foaf:Agent` and `foaf:Document`, for example, should be treated as topics. We can use multiple templates to help the user achieve this goal. Once a user selects a temporary topic node, a hash tree, a template and an XHTML+RDFa page will be generated based on node occurrences. Meanwhile, the relevant NSs are also grouped and displayed on the final page. Thereafter, the generated XHTML+RDFa snippets will be automatically combined into a single snippet.

It is not uncommon that users publish an XHTML+RDFa page using triples from different RDF sources (or in our terminology, from different contexts). We can accommodate this in a way similar to our approach to dealing with multiple topic nodes. Our approach supports federated integration by managing the NSs derived from different RDF documents separately and combining them at the final stage. However, it should be noted that different vocabularies do not necessarily employ the same QName prefix for a given NS. *prefix.cc* (*PCC*) [13] alleviates the issue that RDF documents involve different prefixes indicating the same NS or the same prefix indicating more than one NS by allowing users to look up the collected NSs on *PCC* and vote for their favourite ones. Nevertheless, it is difficult if not impossible to stop people from using ambiguous prefixes.

[13] `http://prefix.cc`

Our approach can automatically detect if a prefix is ambiguous across a set of contexts, and will synthesise new prefixes to ensure disambiguation by generating different prefixes as substitutions. Moreover, many of the NSs in the original RDF context set are unused in the final XHTML+RDFa Web pages. In order to avoid an unnecessary burden on browsers rendering the page, the NSs which are not used in the user's RDF-embedded Web page will be automatically excluded. It is notable that RDFa 1.1 harnesses `@profile` to come over the lengthy declaration of NS prefixes recommended in RDFa 1.0 and this can be also used for avoiding possible ambiguous prefixes to some extent. Figure 4 illustrates the generation of a triple-embedded Web page by combining triples derived from three different Twitter profiles (contexts).

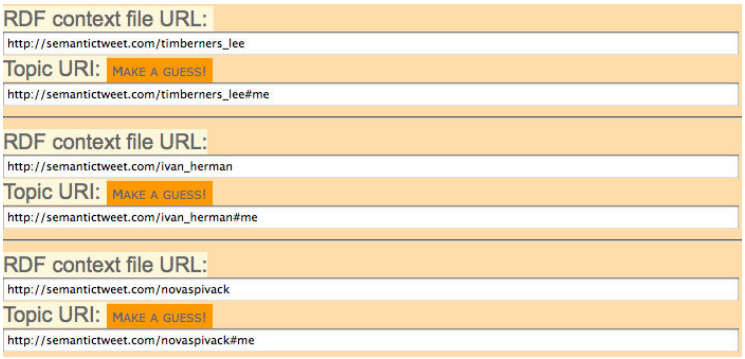


Fig. 4. Screenshot for generating annotation from multiple contexts

Figure 5 illustrates how our approach assists users in creating Web pages annotated with RDF triples derived from different data sources. Users inform RDFa^2 of the target in one or more RDF contexts by providing one or more URLs. These documents will be retrieved on the fly and each of them forms an RDF context. After the topic nodes are selected, triples related to them will be extracted. Finally, the page with RDFa annotation will be sent back to users.

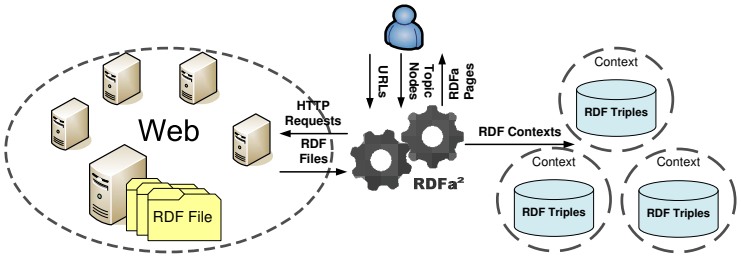


Fig. 5. Context-based federated integration

4.3 Customisation and Template Reuse

One of the primary functions of our approach is to automatically carry out a template-based transformation of RDF to XHTML+RDFa. However, the result of the transformation will almost certainly not be in the precise form required by users, and consequently it is important to allow users to further edit the output. The RDFa^2 interface provides the user with both a rendered preview and the source code of the generated XHTML+RDFa. Users without expertise in RDF(a) can modify the output by clicking and editing elements on the preview page or editing the content in the WYSIWYG way as shown in Figure 2. More experienced users can edit the page source and check its preview but it is recommended that revisions are limited to the text nodes of the page since manually edited RDFa needs revalidation.

When users deal with a great number of RDF documents of the same type (e.g., all of them are FOAF documents), they may have to carry similar or even identical manual revisions for each document processed by RDFa^2. To avoid this unnecessary effort, we provide users with another way of personalising the RDFa-embedded web pages by letting them revise the templates. Each transformation will generate a template and this template will be returned before being applied to the RDF context. A basic template is generated using placeholders of the kind standardly offered by template tools (e.g., FreeMarker [14] applied here). Each placeholder indicates a piece of information which will be extracted during the transformation process (e.g., `personal.firstname` and `personal.lastname` are two placeholders which will be replaced with the first name and the last name of a particular person, respectively). As long as a template is generated, a hash tree that stores the data about the topic nodes is also generated, based on the RDF context: we call this an *intermediate tree*. The structure of the intermediate tree evolved from the structure of the topic tree but is more friendly to templating. Figure 6 shows the excerpt of a generated template that will be used for displaying all the people connected to the selected topic node via `foaf:knows`.

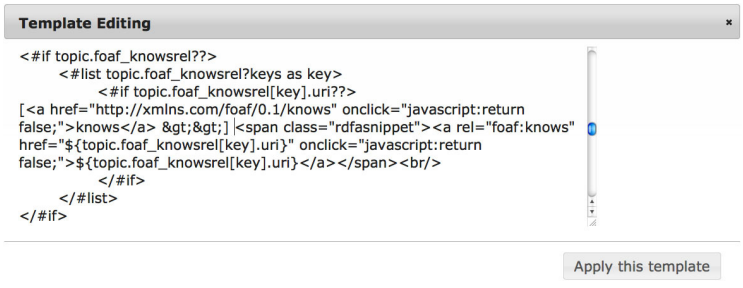


Fig. 6. Excerpt of a generated template for displaying known people

[14] `http://freemarker.org`

The triples taking the topic node as subjects or objects may take literals or other resources as their objects. Both of these two cases have to be taken into consideration before the template is generated. If the object is a literal, it will be enclosed within an HTML tag with `@property` (`@ATTRIBUTE` is used hereafter for denoting a tag's attribute in terms of the XPath syntax) indicating the predicate attached with this object. If the object is a resource, it will be enclosed within by an HTML tag with `@resource` taking this object as its value and `@rel` indicating the predicate attached to this object. As mentioned in Section 3, the text value of this tag node will be the preferred label (if exists) of the resource rather than its URI. This complies with the modelling pattern introduced in [11] as well.

4.4 Self-Adaptability and Reflections on RDF Features

Our approach queries the RDF context using SPARQL with the topic node either given by the user or discovered by the topic recommender semiautomatically, which has been discussed above. The result will be used for replacing the pre-generated placeholders insides templates. In a specific RDF vocabulary, some properties may be defined as functional properties (e.g., `foaf:gender` and `foaf:primaryTopic` in FOAF). Each of them only takes one object or one literal as its value. Other properties (e.g., `foaf:maker` and `foaf:member`) may take more than one object or literal as their values. The SPARQL query results are grouped in terms of properties. With respect to the evolution of an RDF vocabulary, new classes or properties may be involved and some classes or properties may be deprecated. Since templates are created and applied on the fly and always based on the given vocabularies (RDF contexts), the above evolution will be transparent to users. For some of them who want to reuse their templates, their existing templates can be merged with the newly generated ones. A few manual reconciling work on these two kinds of templates might be involved within this process.

According to [1], `@resource` and `@href` can be used for hooking the object of an RDF triple. The value of the former is a URI which is "not intended to be clickable" and normally denotes a non-information resource while the value of the latter is a URI which normally denotes a information resource. The minters of non-clickable URIs need to provide relevant information resources as these URIs' representations [15]. RDFa^2 currently assumes each non-information resource has an informational representation and by clicking it, users will be redirected to another information resource associated with it. Thus, either information resources or non-information resources will be wrapped in `<a>` tags and attached to `@href` rather than `@resource` here. Since `@href` supports only URIs, the object of each RDF triple will not be expressed in CURIE (a generic, abbreviated syntax for expressing URIs) syntax in the final page. With respect to BNodes, the labelling property (if exists) and corresponding value surrounding a specific BNode in the original RDF context will be used as the representation. Nevertheless, users are recommended not to use BNodes when publishing Linked Data on the Web [9].

4.5 Linking Annotations to the LOD Cloud

There is one step to go before RDF triples are injected into Web pages because these embedded triples may otherwise cause provenance and trust issues. RDF statements are focused on describing who said what but statements themselves may or may not be true. Additionally, the licence is another thing that should not be ignored especially when users attempt to reuse data by other data providers. Therefore, the enriched documents need to be associated with provenance information and linked to the Linked Open Data (LOD) Cloud [15]. Here, we use the Vocabulary of Interlinked Datasets (voiD) [2] to describe the relationships between the annotations and the RDF contexts from which the harnessed triples are derived. This vocabulary has been used here due to its simplicity and concision but alternative linked dataset vocabularies could be applied here for the same purpose. Suppose the URI of the topic node is denoted by T_{uri} and the URI of the RDF context (provenance) is denoted by C_{uri}. An XHTML+RDFa snippet will be automatically generated to describe the provenance of T_{uri} as follows:

```
<div about="T_uri" xmlns:void="http://rdfs.org/ns/void#"
   xmlns:dcterms="http://purl.org/dc/terms/">
   <span rel="dcterms:isPartOf">
      <span typeof="void:Dataset">
         <span rel="void:dataDump" resource="C_uri"/>
      </span>
   </span>
</div>
```

5 Experiment and Use-Case Analysis

We experiment with our approach and show the preliminary performance of RDFa^2 , which has been deployed on the Apache Tomcat server installed on a PC with a Pentium®D 3.00GHz × 2 CPU and 1 GB RAM.

Online profiles have been widely used by various Web sites for managing user identification. FOAF is currently one of the most widely used profile vocabulary for RDF on the Web. RDFa^2 can help users inject their FOAF triples into their online profile documents such as homepages. Our experiment first involved asking students who participated in a masters level course on Semantic Web technologies to use RDFa^2 to publish their own profiles (FOAF documents) along with information about their favourite actors/actresses denoted by URIs minted and curated on DBPedia [16] and submit URLs of these documents to Sindice [17]. Fresnel and SPARQLScript were also introduced during the course as alternatives. In total, 64 students participated in this experiment and 60 of them successfully submitted their reports. On a public server, each student has been allocated personal space to store his or her own documents (e.g., the

[15] http://linkeddata.org
[16] http://dbpedia.org/
[17] http://www.sindice.com/main/submit

homepage). By searching documents of type XHTML+RDFa on Sindice with the domain name of the above homepage server as well as students' matriculation numbers, we found that 58 out of 60 students finally published their RDFa profiles and also successfully managed to make Sindice index them. 93.33% of students chose the first topic nodes (at the top of the generated topic-node lists) recommended by our topic-node discovery algorithm as their priority within the process of RDFa snippet generation while two students chose the second topic nodes as their priority. Based on their feedback, RDFa^2 made straightforward the process for generating triple-embedded Web pages from existing RDF data sets and Fresnel as well as SPARQLScript are however more flexible for users with expertise on specific languages as well as RDFa itself to customise pages.

We also collected 324 FOAF documents (without considering dead links declared already on the homepage) from *FOAFBulletinBoard* (*FBB*) [18] and 146 FOAF documents from *W3C RDF Harvester Starting Point* (*WRDFHSP*) [19] respectively. These two sites are separate Wikis for bootstrapping a community in which any users are allowed to contribute FOAF documents collaboratively. Finally we got 149 and 63 valid FOAF documents in total from *FBB* and *WRDFHSP* respectively and republished them with RDFa^2 thereafter. Table 1 shows the results of retrievals of FOAF documents collected from the above two sites.

Table 1. FOAF document retrieval on *FBB* and *WRDFHSP*

Dataset		403	404	406	503	invalid	*UC*	*UKH*	*OOM*	valid
FBB	*N*	6	72	9	1	59	9	18	1	149
	P	2.74%	22.60%	1.37%	.68%	12.33%	6.16%	8.90%	2.05%	43.15%
WRDFHSP	*N*	4	33	2	1	18	9	13	3	63
	P	1.85%	22.22%	2.78%	.31%	18.21%	2.78%	5.56%	.31%	45.99%

In this table, by "invalid", we mean these URLs indicate FOAF documents published in an unrecommended way (e.g., FOAF documents have syntax errors or involve deprecated syntax which can not be accepted by the up-to-date RDF parser). Besides, 403, 404, 406 and 503 denotes the numbers of retrievals that caused HTTP 403, 404, 406 and 503 errors respectively. *UC* denotes the numbers of retrievals that caused unconnected errors and *UKH* denotes the ones caused unknown-host errors. A few FOAF documents contain too many triples to be loaded into our parser and the number of these documents is denoted by *OOM*. *N* and *P* denote the number of retrievals and the corresponding percentage respectively. We see in this table that 54.01% of documents on *FBB* and 56.85% of documents on *WRDFHSP* do not contain valid FOAF information. Due to the space limitation, the time costs of these transformations dedicated to the above two sites are not listed here (see in [3]). On average, 98.58% of valid documents on both sites can be transformed via RDFa^2 within 3 seconds.

[18] `http://wiki.foaf-project.org/w/FOAFBulletinBoard`
[19] `http://esw.w3.org/AnRdfHarvesterStartingPoint`

Besides online profiles, our approach can be also used for republishing RDF vocabularies on normal Web pages. Since there is no central repository of vocabularies on the Semantic Web [20], we collected RDF vocabularies in terms of NSs collected from *Ping The Semantic Web (PTSW)* [21] and *PCC* respectively. At the time of writing this paper, there were 825 NSs recorded by *PTSW*. Since URLs of corresponding vocabularies can not be inferred with the NS URIs(publishers may use rewriting rules to manage the URI of the NS and the URL of the vocabulary document separately), we can only use these NS URIs to do the vocabulary retrievals via HTTP requests as well as content negotiations. Finally we got 249 vocabularies in total and republished them with RDFa^2 afterward. We did the same experiment on *PCC* as well and 349 NS URIs were obtained at the time of writing. We got 165 vocabularies in total and republished them with RDFa^2. Table 2 shows the results of retrievals of RDF vocabularies in terms of names spaces from *PTSW* and *PCC*.

Table 2. RDF vocabulary retrieval in terms of name spaces from *PTSW* and *PCC*

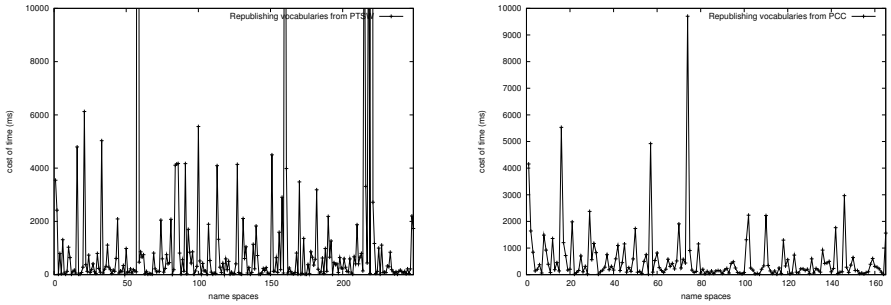
Dataset		400	401	403	404	406	408	500	503
PTSW	*N*	1	8	8	180	14	1	2	33
	P	.12%	.97%	.97%	21.82%	1.70%	.12%	.24%	4.00%
PCC	*N*	-	-	3	35	26	-	-	-
	P	-	-	.86%	10.03%	7.45%	-	-	-
Dataset		invalid	*UC*	*UKH*	*OOM*	valid	-	-	-
PTSW	*N*	258	29	39	3	249	-	-	-
	P	31.27%	3.52%	4.73%	.36%	30.18%	-	-	-
PCC	*N*	108	4	7	1	165	-	-	-
	P	30.95%	1.15%	2.01%	.29%	47.28%	-	-	-

In Table 2, by "invalid", we mean these NS URIs were not valid HTTP URIs or moved temporarily/permanently or indicate vocabularies which were actually not published in the RDF data model or serialised in syntaxes apart from RDF/XML which we just took into consideration in our experiment or published in an unrecommended way (e.g., RDF codes were attached in the comment section of the HTML document or have the error or deprecated syntax which can not be accepted by the RDF parser). Besides, this table contains more columns than Table 1 because more types of HTTP errors occurred within the process of retrieving vocabularies from these two sites. Within the retrieving process, 69.82% vocabularies on *PTSW* and 52.72% vocabularies on *PCC* can not be retrieved by dereferencing their NS URIs.

Figure 7 depicts the costs of time on republishing RDF vocabularies collected in terms of NS URIs from *PTSW* and *PCC* respectively on XHTML pages with embedded RDFa. From this figure, 20 out of 249 vocabularies on *PTSW* as well as 5 out of 165 vocabularies on *PCC* cost around 16ms and there were no results generated after the running of the program. The reason for this is

[20] `http://vocamp.org/wiki/Where_to_find_vocabularies`
[21] `http://pingthesemanticweb.com/`

because these 25 vocabularies in total do not contain any class or property declarations. On average, 93.96% of successfully retrieved vocabularies can be transformed via RDFa^2 within 3 seconds. RDFa^2's performance on dealing with documents containing a large number of triples is related to the employed third party RDF parser and the memory allocated for running the RDFa snippet generation program. It is however not recommended to use RDFa^2 to process large RDF documents since this may lead to Web pages with massive content in the end, which will affect the readability and bring the overhead onto browsers when being rendered.



(a) *PTSW* (b) *PCC*

Fig. 7. Republishing RDF vocabularies on *PTSW* and *PCC*

6 Conclusions

A generic and lightweight approach is proposed to assisting content publishers in generating semantically-enriched hypertext content with triples derived from existing distributed RDF (or RDFa) documents (repositories). The experiment and the use-case analysis show that this approach can help publishers republish their triples in the RDFa serialisation with little human intervention. Nowadays, more and more Linked Data applications have come up and began to employ data from more than one source (or contexts in this paper) and RDFa^2 helps users harness resources from different contexts and potential conflict NSs declarations will be automatically handled. A property of a specific triple could be a topic node as well and a method needs to be carefully designed to synthesise related triples and topic trees. The support in this will be further investigated and integrated in the next step. For other hypertext-friendly formats of embedded metadata such as Microformats [22] or Microdata [23], our approach can be employed as well for generating Web pages with those formats from existing RDF

[22] `http://microformats.org/about`
[23] `http://www.w3.org/TR/microdata`

data sets. At the time of writing, XHTML+RDFa 1.1 [24] and HTML+RDFa 1.1 [25] W3C working drafts were released and have been improved in progress, our goal is to make our approach harness new features compatible with the up-coming W3C recommendations.

References

1. Adida, B., Birbeck, M., McCarron, S., Pemberton, S.: RDFa in XHTML: Syntax and processing, W3C recommendation (2008), http://www.w3.org/TR/rdfa-syntax
2. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets - on the design and usage of voiD, the 'vocabulary of interlinked datasets'. In: Proc. WWW Workshop on LDOW '09 (2009)
3. Bai, X.: Addressing the RDFa Publishing Bottleneck. In: Proc. WWW (Companion Volume) '11. pp. 331–336. ACM Press (2011)
4. Bai, X., Delbru, R., Tummarello, G.: RDF snippets for Semantic Web search engines. In: Proc. OTM '07. vol. 5332, pp. 1304–1318. Springer (2008)
5. Beckett, D., Berners-Lee, T.: Turtle - terse RDF triple language, W3C team submission (2008), http://www.w3.org/TeamSubmission/turtle
6. Beckett, D., McBride, B.: RDF/XML syntax specification (revised), W3C recommendation (2004), http://www.w3.org/TR/REC-rdf-syntax
7. Berners-Lee, T.: Notation 3 specification, W3C design issues (1998), http://www.w3.org/DesignIssues/Notation3.html
8. Berners-Lee, T.: Linked Data (2006), http://www.w3.org/DesignIssues/LinkedData.html
9. Bizer, C., Cyganiak, R., Heath, T.: How to publish Linked Data on the Web (2007), http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial
10. Corlosquet, S., Delbru, R., Polleres, A., Decker, S.: Produce and consume Linked Data with Drupal! In: Proc. ISWC '09. vol. 5823, pp. 763–778. Springer (2009)
11. Dodds, L., Davis, I.: Linked Data patterns - a pattern catalogue for modelling, publishing, and consuming Linked Data (2010), http://patterns.dataincubator.org/book
12. Grant, J., Beckett, D., McBride, B.: RDF test cases, W3C recommendation (2004), http://www.w3.org/TR/rdf-testcases
13. Hepp, M.: GoodRelations: An ontology for describing products and services offers on the Web. In: Proc. EKAW '08. vol. 5268, pp. 332–347. Springer (2008)
14. Hepp, M., García, R., Radinger, A.: RDF2RDFa: Turning RDF into snippets for copy-and-paste. In: Proc. Posters and Demonstrations Track on ISWC '09 (2009)
15. Lewis, R.: Dereferencing HTTP URIs (2007), http://www.w3.org/2001/tag/doc/httpRange-14/HttpRange-14.html
16. Pietriga, E., Bizer, C., Karger, D., Lee, R.: Fresnel: A browser-independent presentation vocabulary for RDF. In: Proc. ISWC '06. vol. 4273, pp. 158–171. Springer (2006)
17. Zhang, X., Cheng, G., Qu, Y.: Ontology summarization based on RDF sentence graph. In: Proc. WWW '07. pp. 707–716. ACM Press (2007)

[24] http://www.w3.org/TR/2010/WD-xhtml-rdfa-20101109

[25] http://www.w3.org/TR/rdfa-in-html